CLAIMS:

What is claimed is:

1. A method of verifying the integrity of unauthenticated code, comprising:

receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code;

receiving the unauthenticated code;

generating a second hash value of the unauthenticated code;

comparing the first hash value and the second hash value; and

verifying the integrity of the unauthenticated code if the first hash value and the second hash value match.

2. The method of claim 1, wherein the automatically authenticated code is compiled Java code and wherein the unauthenticated code is native code.

3. The method of claim 1, wherein the automatically authenticated code is a Java application or applet and wherein the unauthenticated code is a dynamically linked library.

4. The method of claim 1, wherein the first hash value is obtained using a hashing function and wherein generating a second hash value of the unauthenticated code includes using the same hashing function as was used to obtain the first hash value.

5. The method of claim 4, wherein the hashing function is identified based on information stored in the automatically authenticated code.

6. The method of claim 1, further comprising:

executing the automatically authenticated code using a virtual machine; and

sending a request to a server from which the automatically authenticated code was received, the request being for the unauthenticated code.

7. The method of claim 1, wherein, if the first hash value and the second hash value do not match, the method further comprises:

receiving the unauthenticated code again;

generating a third hash value of the unauthenticated code; and

comparing the first hash value and the third hash value.

8. The method of claim 7, wherein if the third hash value and the first hash value do not match, the method further comprises:

comparing the second hash value and the third hash value; and

if the second hash value and the third hash value

match, determining that the unauthenticated code has been corrupted intentionally.

9. The method of claim 8, wherein if the second hash value and the third hash value do not match, it is determined that the unauthenticated code has been corrupted unintentionally.

10. The method of claim 1, wherein the method is implemented in a virtual machine associated with a web browser on a client device.

11. An apparatus for verifying the integrity of unauthenticated code, comprising:

a virtual machine; and

an unauthenticated code verification element, wherein the virtual machine receives automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code, and receives the unauthenticated code, and wherein

the unauthenticated code verification element generates a second hash value of the unauthenticated code, compares the first hash value and the second hash value, and verifies the integrity of the unauthenticated code if the first hash value and the second hash value match.

12. The apparatus of claim 11, wherein the automatically authenticated code is compiled Java code and wherein the

unauthenticated code is native code.

13. The apparatus of claim 11, wherein the automatically authenticated code is a Java application or applet and wherein the unauthenticated code is a dynamically linked library.

14. The apparatus of claim 11, wherein the first hash value is obtained using a hashing function and wherein the unauthenticated code verification element generates a second hash value of the unauthenticated code using the same hashing function as was used to obtain the first hash value.

15. The apparatus of claim 14, wherein the hashing function is identified by the unauthenticated code verification element based on information stored in the automatically authenticated code.

16. The apparatus of claim 11, wherein the virtual machine executes the automatically authenticated code and sends a request to a server from which the automatically authenticated code was received, the request being for the unauthenticated code.

17. The apparatus of claim 11, wherein, if the first hash value and the second hash value do not match, the virtual machine receives the unauthenticated code again, the unauthenticated code verification element generates a third hash value of the

unauthenticated code and compares the first hash value and the third hash value.

18. The apparatus of claim 17, wherein if the third hash value and the first hash value do not match, the unauthenticated code verification element compares the second hash value and the third hash value and, if the second hash value and the third hash value match, determines that the unauthenticated code has been corrupted intentionally.

19. The apparatus of claim 18, wherein if the second hash value and the third hash value do not match, the unauthenticated code verification element determines that the unauthenticated code has been corrupted unintentionally.

20. The apparatus of claim 11, wherein the virtual machine and the unauthenticated code verification element are associated with a web browser on a client device.

21. A computer program product in a computer readable medium for verifying the integrity of unauthenticated code, comprising:

first instructions for receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code;

second instructions for receiving the unauthenticated

code;

third instructions for generating a second hash value of the unauthenticated code;

fourth instructions for comparing the first hash value and the second hash value; and

fifth instructions for verifying the integrity of the unauthenticated code if the first hash value and the second hash value match.

22. The computer program product of claim 21, wherein the automatically authenticated code is compiled Java code and wherein the unauthenticated code is native code.

23. The computer program product of claim 21, wherein the automatically authenticated code is a Java application or applet and wherein the unauthenticated code is a dynamically linked library.

24. The computer program product of claim 21, wherein the first hash value is obtained using a hashing function and wherein the third instructions for generating a second hash value of the unauthenticated code include instructions for using the same hashing function as was used to obtain the first hash value.

25. The computer program product of claim 24, further comprising instructions for identifying the hashing function based on information stored in the automatically authenticated

code.

26. The computer program product of claim 21, further comprising:

sixth instructions for executing the automatically authenticated code using a virtual machine; and

seventh instructions for sending a request to a server from which the automatically authenticated code was received, the request being for the unauthenticated code.

27. The computer program product of claim 21, further comprising:

sixth instructions for receiving the unauthenticated code again, if the first hash value and the second hash value do not match;

seventh instructions for generating a third hash value of the unauthenticated code; and

eighth instructions for comparing the first hash value and the third hash value.

28. The computer program product of claim 27, further comprising:

ninth instructions for comparing the second hash value and the third hash value, if the third hash value and the first hash value do not match; and

tenth instructions for determining that the unauthenticated code has been corrupted intentionally, if the

second hash value and the third hash value match.

29. The computer program product of claim 28, further comprising eleventh instructions for determining that the unauthenticated code has been corrupted unintentionally, if the second hash value and the third hash value do not match.